Information Sciences 180 (2010) 1802–1816

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Timer-based composition of fault-containing self-stabilizing protocols

Yukiko Yamauchi [a,*], Sayaka Kamei [b,1], Fukuhito Ooshita [c,2], Yoshiaki Katayama [d,3], Hirotsugu Kakugawa [c,2], Toshimitsu Masuzawa [c,2]

[a] Graduate School of Information Science, Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara 630-0192, Japan
[b] Department of Information Engineering, Graduate School of Engineering, Hiroshima University, 1-4-1 Kagamiyama, Higashi Hiroshima, Hiroshima 739-8527, Japan
[c] Graduate School of Information Science and Technology, Osaka University, 1-5, Yamadaoka, Suita, Osaka 565-0871, Japan
[d] Graduate School of Computer Science and Engineering, Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya 466-8555, Japan

ARTICLE INFO

Article history:
Received 21 November 2008
Received in revised form 1 October 2009
Accepted 4 October 2009

Keywords:
Distributed system
Fault tolerance
Self-adaptability
Self-stabilization
Fault-containment
Hierarchical composition
Timer
Synchronizer

ABSTRACT

One of the desired properties of distributed systems is self-adaptability against faults. Self-stabilizing protocols provide autonomous recovery from any finite number of transient faults. However, in practice, catastrophic faults rarely occur, while small-scale faults are more likely to occur. Fault-containing self-stabilizing protocols promise not only self-stabilization but also containment of the effect of small-scale faults, *i.e.*, they promise quick recovery and small effect for small-scale faults. Hierarchical composition of self-stabilizing protocols is expected to ease the design of new self-stabilizing protocols. However, existing composition techniques for self-stabilizing protocols cannot preserve the fault-containment property of source protocols. In this paper, we propose a novel timer-based hierarchical composition of fault-containing self-stabilizing protocols that preserves the fault-containment property of source protocols. To implement timers, we propose a local neighborhood synchronizer that synchronizes limited number of processes during a short time after a fault without involving the entire network into the synchronization. The proposed composition technique facilitates the design of new fault-containing self-stabilizing protocols and enhances the reusability of existing fault-containing self-stabilizing protocols.

1. Introduction

A distributed system consists of processes communicating with each other by communication links. It is expected to add useful properties to distributed systems, *e.g.*, performance, availability, and scalability. Large scale networks that consist of a large number of processes have became popular such as the Internet and peer-to-peer networks. As the number of processes in a distributed system grows, the distributed system becomes more prone to faults. The effect of faults may spread over the entire network due to the communication among processes and the whole system may be disrupted. This is the reason why fault-tolerance is the major concern when we design distributed systems.

Self-stabilization provides autonomous adaptability against any finite number of *transient faults* (*e.g.*, memory crash at processes). Starting from an arbitrary initial configuration, a self-stabilizing protocol converges to a legitimate configuration where the protocol satisfies its specification. A self-stabilizing protocol guarantees autonomous adaptability by considering

* Corresponding author. Tel.: +81 743 72 5253; fax: +81 743 72 5259.
E-mail addresses: y-yamauchi@is.naist.jp (Y. Yamauchi), s-kamei@se.hiroshima-u.ac.jp (S. Kamei), f-oosita@ist.osaka-u.ac.jp (F. Ooshita), katayama@nitech.ac.jp (Y. Katayama), kakugawa@ist.osaka-u.ac.jp (H. Kakugawa), masuzawa@ist.osaka-u.ac.jp (T. Masuzawa).
[1] Tel.: +81 82 424 7685; fax: +81 82 422 7195.
[2] Tel.: +81 6 6879 4118; fax: +81 6 6879 4119.
[3] Tel.: +81 52 735 5576; fax: +81 52 735 5465.

0020-0255/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2009.10.003

the configuration obtained by the last fault as an initial configuration. Since self-stabilization was first introduced by Dijkstra [6], many self-stabilizing protocols have been designed for many problems, *e.g.*, for graph problems [1,4,18,19,22,27], other problems [3,9,20,23,25]. Though self-stabilization is derived from theoretical studies, many researchers applied the notion to real networks, *e.g.*, sensor networks, mobile ad-hoc networks [17,21]. (Good surveys are available in [7,10,26].) Although self-stabilization promises fault tolerance against any finite number of transient faults, it promises nothing during the stabilization and the effect of a small-scale fault can spread over the entire network. However, it is generally expected that the system recovers quickly with small effect from a small-scale fault. Therefore, it is practically useful to restrict the spread of the effect of small-scale faults. This is because catastrophic faults rarely occur in practice and small-scale faults are more likely to occur.

When a fault corrupts f processes in a legitimate configuration, we call the obtained configuration an *f-faulty configuration*. Ghosh et al. proposed the notion of fault-containment [11,12]. An *f-fault-containing self-stabilizing protocol* promises self-stabilization against large scale faults and fault-containment against small-scale faults [11,12]. (We denote it by an *f-fault-containing protocol*.) An f-fault-containing protocol guarantees that starting from any f'-faulty configuration ($f' \leqslant f$), during the recovery to a legitimate configuration, both the recovery time and the number of processes affected by the fault are proportional to f or less. So, fault-containment improves the adaptability of self-stabilization against small-scale faults. After the notion is introduced, many fault-containing protocols have been proposed [5,13–15,24].

Hierarchical composition of multiple protocols facilitates the design of new protocols. In a hierarchical composition of two (or more) protocols, the output of one protocol (called *the lower protocol*) is used as the input to the other (called *the upper protocol*), and the obtained protocol provides the output of the upper protocol when given the input to the lower protocol. Fig. 1 shows an example of hierarchical composition of two protocols such that the lower protocol is a spanning tree construction protocol for an arbitrary network and the upper protocol is a token circulation protocol for an arbitrary tree. The spanning tree constructed by the lower protocol is used as the input by the upper protocol, and the composite protocol is a token circulation protocol for an arbitrary network.

One of the most commonly used hierarchical composition technique for self-stabilizing protocols is *fair composition* [9]. Fair composition executes two (or more) self-stabilizing protocols in parallel and promises self-stabilization of the composite protocol. Starting from an arbitrary initial configuration, the lower protocol converges to its legitimate configuration first, and after that, the upper protocol converges with a correct input from the lower protocol. Fair composition greatly reduces the complication of designing self-stabilizing protocols, and many self-stabilizing protocols are designed by using fair composition. In [9], a mutual exclusion protocol on a spanning tree construction protocol is proposed. In [22], an approximation protocol for minimum connected dominating set problem is designed on a maximal independent set protocol. By using fair composition with spanning tree construction protocols, many self-stabilizing protocols designed for tree networks can be executed on arbitrary networks, *e.g.*, PIF (Propagation of Information and Feedback) protocol in [3], synchronizers in [20], and agent traversal protocol in [25]. In [17], a TDMA slot assignment protocol for sensor networks is proposed which consists of a naming protocol, a maximal independent set protocol, a coloring protocol, and a slot assignment protocol.

Unfortunately, fair composition of fault-containing protocols does not preserve the fault-containment property of source protocols. This is because the parallel execution of the source protocols allows the upper protocol to execute on an incorrect output of the lower protocol. Consider a fair composition of f_1-fault-containing protocol P_1 and f_2-fault-containing protocol P_2. Suppose that a fault corrupts the output variables of the lower protocol P_1 at f processes ($f \leqslant \min\{f_1, f_2\}$). During the recovery of P_1, the upper protocol P_2 is also executed in parallel. During the recovery of P_1, processes around each faulty process may change their states in P_1. Because the changes of the values of their output variables of P_1 implies the changes of the input to P_2, if the number of such affected processes in P_1 becomes greater than f_2, P_2 cannot guarantee fault-containment. Even when the number of such affected processes in P_1 is smaller than f_2, if these processes change their outputs of P_1 repeatedly, P_2 cannot promise fault-containment because a fault-containing protocol assumes that the input does not change during the recovery.

Gouda et al. proposed *adaptive programming* for the systems with input changes [16]. They proposed hierarchical composition technique for adaptive protocols that forces the lower protocol to execute first so that it provides the stable input to the upper protocol. Their hierarchical composition just checks whether a process has to execute the lower protocol (*i.e.*, whether it has an enabled process in the lower protocol), and only when it does not have to execute the lower protocol, the process can execute the upper protocol. However, hierarchical composition preserves the self-stabilization property of source protocols, but not the fault-containment property of source protocols. The problem is that we cannot guarantee the recovery of the lower protocol by checking the existence of enabled guards in the lower protocol. This is one of the key observations to achieve the composition of fault-containing protocols.

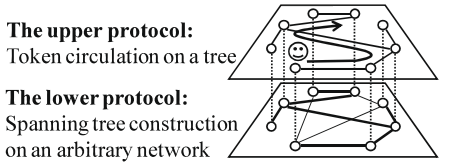

Fig. 1. An example of hierarchical composition.